(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 198 26 826 A 1**

(51) Int. Cl.[6]: G 06 F 9/30

(21) Aktenzeichen: 198 26 826.2
(22) Anmeldetag: 16. 6. 98
(43) Offenlegungstag: 15. 7. 99

DE 198 26 826 A 1

Mit Einverständnis des Anmelders offengelegte Anmeldung gemäß § 31 Abs. 2 Ziffer 1 PatG

(71) Anmelder:
Siemens AG, 80333 München, DE

(72) Erfinder:
He, Ping, Dr.-Ing., 81369 München, DE; Osterholzer, Rudolf, Dipl.-Ing. (FH), 82008 Unterhaching, DE

(56) Entgegenhaltungen:
WO 97 13 194 A1

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zum Decodieren und Ausführen von Befehlen in einem RISC-Prozessor

(57) Die vorliegende Erfindung bezieht sich auf ein Verfahren zum Decodieren und Ausführen von Befehlen in einem RISC-Prozessor, wobei jeder Befehl eine vorgegebene gleiche Befehlslänge besitzt und ein Opcodefeld sowie zumindest zwei Operandenfelder aufweist. Die Wahl eines bestimmten Befehlsformates mit vorgegebener Befehlslänge gestattet sowohl eine Erhöhung des Datendurchsatzes durch den Prozessor wie auch eine Verbesserung der Flexibilität des Befehlssatzes.

DE 198 26 826 A 1

BNSDOCID: <DE__19826826A1_I_>

Beschreibung

Die Erfindung bezieht sich auf ein Verfahren zum Decodieren und Ausführen von Befehlen in einem RISC-Prozessor, wobei jeder Befehl eine vorgegebene gleiche Befehlslänge besitzt und ein Opcodefeld, zumindest zwei Operandenfelder zum Adressieren von Operanden mit variabler Länge sowie ein Feld, welches die Länge der Operanden festlegt, aufweist.

Die herkömmlichen Befehlssätze der am Markt üblichen Prozessoren lassen sich allgemein in zwei Gruppen unterteilen. Die eine Gruppe besitzt einen großen Befehlsumfang und arbeitet mit einer flexiblen Befehlslänge, weshalb sie für eine Optimierung des Datendurchsatzes in einem RISC-Prozessor mit Hilfe von Prefetch- oder Pipelining-Operationen weniger geeignet ist. Die andere Gruppe arbeitet zwar mit festen Befehlslängen, die Befehlssätze dieser Gruppe verfügen allerdings nur über einen eingeschränkten bzw. wenig flexiblen Befehlsumfang.

Es ist deshalb die Aufgabe der Erfindung, ein Verfahren zum Dekodieren und Ausführen von Befehlen in einem RISC-Prozessor bereitzustellen, welches einen optimierten Datendurchsatz bei gleichzeitig großer Flexibilität der Prozessorbefehle gewährleistet.

Diese Aufgabe wird durch das in Anspruch 1 beanspruchte Verfahren gelöst.

Gemäß der Erfindung wird die Aufgabe in der Weise gelöst, daß zwar einerseits jeder Befehl eine vorgegebene gleiche Befehlslänge besitzt, daß aber andererseits auch Operanden mit variabler (flexibler) Länge adressiert werden können.

Bei dem Verfahren gemäß der vorliegenden Erfindung wird ein Befehlssatz verwendet, bei dem jeder Befehl ein Opcodefeld und zumindest zwei Operandenfelder zum Adressieren von Operanden sowie ein Feld, welches die Länge der Operanden-festlegt. Das Opcodefeld beinhaltet den Operator des Befehls, d. h. es enthält in codierter Form die Art der Operandenverknüpfung.

Durch die Vorgabe einer festen Befehlslänge wird es der Execution Unit des Prozessors ermöglicht, einfache Prefetch- oder Pipeliningoperationen auszuführen, wodurch der Datendurchsatz durch den Prozessor erhöht wird.

Gleichzeitig gestattet das Opcodefeld eine freie Zuordnung, welches Operandenfeld das Ziel und welches Operandenfeld die Quelle einer jeweiligen Operandenverknüpfung darstellt; aufgrund dieser freien Zuordnung ist der Befehlssatz gleichzeitig sehr flexibel.

Schließlich sei es als Vorteil erwähnt, daß das verwendete Befehlsformat mit fester Befehlslänge und definierten Opcode- und Operandenfeldlängen die Grundlage für eine einfache und übersichtliche Befehlsstruktur bietet. Obwohl die Längen der Operandenfelder jeweils fest definiert sind, gelingt es über verschiedene Adressierungsarten, Operanden unterschiedlicher Länge zu adressieren.

Gemäß einer vorteilhaften Ausgestaltung des Verfahrens ist die Befehlslänge auf 32 Bit fest vorgegeben.

Ein Register in einem ersten Operandenfeld bietet den Vorteil, daß die Gesamtbefehlslänge auch dann konstant gehalten werden kann, wenn erste Operanden mit variabler Länge, z. B. 1, 2, 4 oder 16 Byte, adressiert werden sollen.

Eine Registeradressierung ist insofern von Vorteil, als daß sie verschiedene Adressierungsarten ermöglicht, genauer gesagt neben einer direkten auch eine indirekte Registeradressierung, gegebenenfalls auch mit Postinkrement.

Gemäß einer weiteren vorteilhaften Ausgestaltung steht im zweiten Operandenfeld entweder ein Register, eine absolute Speicheradresse oder eine Datenkonstante.

Wird der zweite Operand über ein Register adressiert, so ist er ebenfalls direkt, indirekt oder indirekt mit Postinkrement adressierbar. Im Falle einer Registeradressierung können auch zweite Operanden mit einer variablen Länge von z. B. 1, 2, 4 oder 16 Byte adressiert werden.

Schließlich umfaßt gemäß einer vorteilhaften Ausgestaltung der Erfindung die absolute Speicheradresse oder die Datenkonstante jeweils 16 Bit.

Es folgt eine detaillierte Beschreibung eines bevorzugten Ausführungsbeispiels der vorliegenden Erfindung unter Bezugnahme auf die folgenden Zeichnungen. Dabei zeigt:

Fig. 1 ein Befehlsformat, welches bei den erfindungsgemäßen Verfahren verwendet wird;

Fig. 2 eine nichtvollständige Tabelle als beispielhaften Inhalt des Opcodefeldes;

Fig. 3 eine 2-Bitcodierung der Operandenlängen als beispielhaften Inhalt für das Feld, das die Operandenlängen festlegt;

Fig. 4 die Struktur eines 6-Bit breiten ersten Operandenfeldes im Befehlsformat nach **Fig.** 1;

Fig. 5 eine 2-Bitcodierung der auswählbaren Registeradressierungsarten als beispielhaften Inhalt der höchstwertigen 2-Bit des ersten Operandenfeldes der **Fig.** 4;

Fig. 6 eine 4-Bitcodierung aller verfügbaren Register als beispielhaften Inhalt der niederwertigsten 4-Bit des ersten Operandenfeldes der **Fig.** 4;

Fig. 7 eine Grobstruktur des 18-Bit breiten zweiten Operandenfeldes im Befehlsformat nach **Fig.** 1;

Fig. 8 eine 2-Bitcodierung auswählbarer Adressierungarten für den zweiten Operanden als beispielhaften Inhalt der höchstwertigen 2-Bit des zweiten Operandenfeldes;

Fig. 9 eine Teilstruktur des zweiten Operandenfeldes der **Fig.** 7, wenn eine Registeradressierung des zweiten Operanden vorgesehen ist;

Fig. 10 eine 2-Bitcodierung aller Registeradressierungsarten als beispielhaften Inhalt der Bits **14** und **15** des zweiten Operandenfeldes nach **Fig.** 9;

Fig. 11 eine 4-Bitcodierung aller im Supervisormode selektierbaren Register als beispielhaften Inhalt der Bits **10** bis **13** des zweiten Operandenfeldes nach **Fig.** 9;

Fig. 12 eine 1-Bitcodierung aller im Supervisormode auswählbaren Usertasks als möglichen Inhalt des Bits **9** des zweiten Operandenfeldes nach **Fig.** 9;

Fig. 13 eine 2-Bitcodierung aller im Supervisormode selektierbaren Kontrollregister als möglichen Inhalt der Bits **7** und **8** im zweiten Operandenfeld nach **Fig.** 9;

Fig. 14 eine Teilstruktur des zweiten Operandenfeldes, wenn eine direkte Adressierung des zweiten Operanden über eine absolute Memory-Adresse vorgesehen ist; und

Fig. 15 eine Teilstrukturierung des zweiten Operandenfeldes, wenn eine Adressierung des zweiten Operanden als Datenkonstante vorgesehen ist.

Gemäß dem bevorzugten Ausführungsbeispiel der vorliegenden Erfindung weist jeder Befehl eines RISC-Prozessors vorzugsweise das in **Fig.** 1 dargestellte Befehlsformat auf. Es hat insgesamt eine Länge von 32 Bit und ist in vier Felder mit jeweils vorgegebener Bitlänge aufgeteilt.

Der Bereich der höherwertigeren Bits wird von einem 6-Bit breiten Opcodefeld belegt, das von Bit **31**, dem höchstwertigen Bit, bis einschließlich Bit **25** reicht. Rechtsbündig an das Opcodefeld schließt sich ein 2-Bit breites Feld (Size-Feld), bestehend aus Bit **24** und Bit **23** des Befehlsformates an, welches die Länge eines Operanden definiert. Dem Size-Feld rechts benachbart ist ein erstes Operandenfeld mit einer Länge von 6 Bit, das von Bit **22** bis Bit **18** einschließlich reicht. Schließlich definiert der untere Bereich des Befehlsformates, der sich von Bit **17** bis Bit **0**, dem niederwertigsten Bit erstreckt, ein zweites Operandenfeld.

Im Folgenden werden die einzelnen Felder des erlin-

dungsgemäß verwendeten Befehlsformates näher beschrieben, wobei gegebenenfalls eine noch weitergehende Untergliederung erfolgt.

Das Opcodefeld definiert zum einen in codierten Form die Art der Operandenverknüpfung, d. h. es legt fest, ob die Operanden logisch, arithmetisch oder in anderer Weise verknüpft werden. Darüber hinaus legt es eine Zuordnung fest, welcher der durch die beiden Operandenfelder definierten Operanden das Ziel oder die Quelle für eine vorher definierte Operandenverknüpfung darstellt. Die Frage der Zuordnung ist insbesondere bei Load/Store-Operationen von Bedeutung.

Fig. 2 listet beispielhaft einige Operandenverknüpfungen (arithmetische-, logische oder Shiftoperationen) zusammen mit ihren zugehörigen 6-Bit breiten Codierungen (Opcodes) auf.

Fig. 3 zeigt 2-Bit breite Binärcodes, die als Inhalt des Sizefeldes beispielhafte Operandenlängen definieren.

Gemäß dem hier beschriebenen Ausführungsbeispiel kann ein erster Operand nur über ein Register adressiert werden. Andere Adressierungsarten sind natürlich ebenfalls möglich.

Fig. 4 zeigt eine weitere Unterteilung des 6-Bit breiten ersten Operandenfeldes aus **Fig.** 1. Genauer gesagt, bezeichnen die zwei höherwertigeren Bits **23** und **22** des ersten Operandenfeldes die verwendete Registeradressierungsart, während die vier niederwertigeren Bits **21** bis **18** eine Codierung aller gemäß **Fig.** 6 selektierbaren Register beinhalten.

Gemäß **Fig.** 5 stehen in einem User-Mode insgesamt drei Adressierungsarten zur Auswahl. User-Mode bedeutet, daß nur die einem Softwaremodul (Task) zugeordneten Register adressiert werden können. Die drei zur Auswahl stehenden Adressierungsarten sind direkte und indirekte Registeradressierung sowie indirekte Registeradressierung mit Postinkrement. Dabei bedeutet eine indirekte Registeradressierung, daß der Operand über eine im Register stehende Adresse eines Speichers adressiert wird. Dem gegenüber bedeutet indirekte Registeradressierung mit Postinkrement, daß im Anschluß an einen Befehl, die Operandenadresse in dem Register um die Operandenlänge (hier 1, 2, 4 oder 16 Byte), wie sie im Sizefeld codiert vorliegt, erhöht wird.

Fig. 7 zeigt die Grobstruktur des 18-Bit breiten zweiten Operandenfeldes. Der 2-Bitcode in den höherwertigen Bits **16** und **17** des zweiten Operandenfeldes legt fest, ob es sich bei dem Inhalt dieses Feldes um eine Registeradresse, eine absolute Speicheradresse oder um eine Datenkonstante handelt. Eine Codierung der beispielhaften Feldinhalte zeigt **Fig.** 8.

Die weitere Feinstrukturierung der niederwertigen 16 Bit des zweiten Operandenfeldes wird durch den in den höherwertigen Bits **16** und **17** codierten sog. Operandenmode festgelegt.

Zunächst wird der Fall betrachtet, daß der Operandenmode eine Registeradressierung für den zweiten Operanden festlegt; dann weisen die Bits **15** bis **0** des zweiten Operandenfeldes die in **Fig.** 9 dargestellte Teilstruktur auf.

Gemäß **Fig.** 9 zeigen zunächst die Bits **15** und **14** die selektierte Registeradressierungsart an. Dabei stehen gemäß **Fig.** 10 im User-Mode dieselben Registeradressierungsarten wie für das erste Operandenfeld zur Verfügung, nämlich direkte und indirekte Registeradressierung sowie indirekte Registeradressierung mit Postinkrement. Darüber hinaus steht als vierte Möglichkeit eine direkte Registeradressierung im Supervisor-Mode zur Verfügung. Supervisor-Mode bedeutet, daß gemäß dem Wert von Bit **9**, dessen beispielhafte Inhalte in **Fig.** 12 dargestellt sind, spezielle Softwaremodule (Usertasks) ausgewählt werden können und das darüber hinaus durch Wahl der Binärwerte an den Bit-Positionen **7** und **8** (siehe **Fig.** 13) auf spezielle Kontrollregister zugegriffen werden kann.

Bit **9** (User-Task-Selekt-Feld) sowie die Bits **8** und **7** (Kontrollregisterfeld) werden nur im Supervisor-Mode, nicht aber im User-Mode ausgewertet.

Für eine Registeradressierung des zweiten Operanden stehen im User-Mode neben denselben Registeradressierungsarten auch nahezu alle Register wie für den ersten Operanden zur Verfügung (siehe **Fig.** 6). **Fig.** 11 zeigt eine Binärcodierung aller über die Bits **13** bis **10** im zweiten Operandenfeld selektierbaren Register im Supervisor-Mode. **Fig.** 11 unterscheidet sich von **Fig.** 6 lediglich durch die Möglichkeit zur Auswahl (Zusatzadressierung) eines Kontrollregisters, das an die Stelle des Stackpointerregisters aus **Fig.** 6 getreten ist. Wenn im Supervisor-Mode eine entsprechende Codierung der Bits **10** bis Bit **13** im zweiten Operandenfeld diese Zusatzadressierung des Kontrollregisters auswählt, dann stehen über die Bits **7** und **8** (Kontrollregisterfeld) weitere Spezialregister, wie Stackpointer, Programmcounter oder Instructionregister gemäß **Fig.** 13 zur Adressierung zur Verfügung.

Im Falle einer Registeradressierung kann sowohl der erste wie auch der zweite Operand eine variable Länge von 1, 2, 4 oder 16 Byte aufweisen. Insbesondere bei einer direkten Registeradressierung wurde ein Operand mit einer Operandenlänge von 16 Byte auf vier 32 Bit-Register, z. B. die Register R0 bis R3 aufgeteilt.

Wie bereits oben angedeutet und in **Fig.** 8 dargestellt, kann der zweite Operand nicht nur über ein Register adressiert werden, sondern auch über die Angabe einer 16-Bit breiten absoluten Speicheradresse. In diesem Fall belegt die 16-Bit breite Speicheradresse die niederwertigsten 16 Bit des zweiten Operandenfeldes, also Bit **0** bis Bit **15**. Mit der 16-Bit breiten Speicheradresse können bekanntlich 216 Adressen direkt im Speicher adressiert werden. Für eine Adressierung des darüber hinaus gehenden Speicherbereiches wird ausschließlich eine indirekte Registeradressierungsart verwendet.

Schließlich bietet das zweite Operandenfeld auch die Möglichkeit, den zweiten Operanden als 16-Bit-Datenkonstante zu speichern. Eine entsprechende Festlegung erfolgt gemäß **Fig.** 8 in den Bits **16** und **17**, wobei dann die eigentliche Datenkonstante Bit **0** bis Bit **15** belegt. Eine 32-Bit-Datenkonstante kann mit zwei Befehlen in einem Register (mit jeweils einer 16-Bit-Konstante) zusammengefügt werden.

Patentansprüche

1. Verfahren zur Decodierung und Ausführung von Befehlen in einem RISC-Prozessor, wobei jeder Befehl eine vorgegebene gleiche Befehlslänge besitzt und ein Opcodefeld, zumindest zwei Operandenfelder zum Adressieren von Operanden mit variabler Länge sowie ein Feld, welches die Länge der Operanden festlegt, aufweist, mit folgenden Schritten:

   - Dekodieren des Inhaltes des Opcodefeldes, um die Art der Operandenverknüpfung und/oder eine Zuordnung, welches Operandenfeld das Ziel und welche Operandenfeld die Quelle darstellt, festzulegen;
   - Dekodieren des Inhaltes des Feldes, in dem die Länge der Operanden festgelegt ist;
   - Dekodieren der Inhalte der Operandenfelder, um die Operanden zu ermitteln;
   - Ausführen der Operandenverknüpfung auf die Operanden.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß als feste Befehlslänge 32 Bit vorgegeben ist.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß in dem ersten Operandenfeld ein Register steht.

4. Verfahren nach Anspruch 3, dadurch gekennzeichnet, daß ein erster Operand über das Register entweder direkt, indirekt oder indirekt mit Postinkrement adressierbar ist.

5. Verfahren nach Anspruch 3 oder 4, dadurch gekennzeichnet, daß in dem zweiten Operandenfeld ein Register, eine absolute Speicheradresse oder eine Datenkonstante steht.

6. Verfahren nach Anspruch 5, dadurch gekennzeichnet, daß ein zweiter Operand über das Register entweder direkt, indirekt oder indirekt mit Postinkrement adressierbar ist.

7. Verfahren nach Anspruch 5 oder 6, dadurch gekennzeichnet, daß die absolute Speicheradresse 16 Bit umfaßt.

8. Verfahren nach einem der Ansprüche 5 bis 7, dadurch gekennzeichnet, daß die Datenkonstante 16 Bit umfaßt.

9. Verfahren nach einem der Ansprüche 5 bis 8, dadurch gekennzeichnet, daß im Falle einer Registeradressierung die Länge für den ersten und/oder zweiten Operanden wenigstens ein Byte beträgt.

10. Verfahren nach Anspruch 9, dadurch gekennzeichnet, daß die Operandenlänge 16 Byte beträgt.

Hierzu 4 Seite(n) Zeichnungen

Nummer: DE 198 26 826 A1
Int. Cl.[6]: G 06 F 9/30
Offenlegungstag: 15. Juli 1999

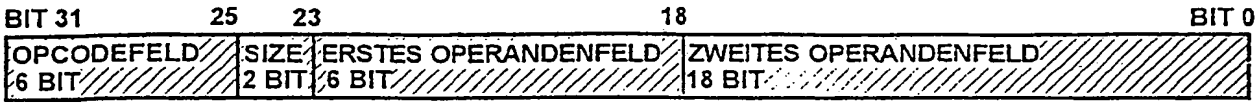

Fig. 1

BIT 31	BIT 25
6 BIT OPCODE	BINÄR CODIERT
DATEN- OPERATION	
000000	LOAD
000001	STORE
...	
ARITHM. OPERATION	
011000	ADDITION
011100	SUBTRAKTION
...	
LOGISCHE OPERATION	
100000	AND
100001	ODER
...	
SHIFT & ROTATE	
110000	LOGICAL SHIFT LEFT
110001	LOGICAL SHIFT RIGHT
...	

Fig. 2

Nummer: DE 198 26 826 A1
Int. Cl.[6]: G 06 F 9/30
Offenlegungstag: 15. Juli 1999

BIT 24		BIT 23
2 BIT		SIZE (BINÄR CODIERT)
0	0	8 BIT
0	1	16 BIT
1	0	32 BIT
1	1	4 x 32 BIT

Fig. 3

BIT 23	BIT 22	BIT 21	BIT 18
REGISTER- ADERESSIERUNGS- MODUS 2 BIT		REGISTER- SELECT- CODE 4 BIT	

Fig. 4

2 BIT		REGISTER- ADRESSIERUNGS- MODE (BINÄR CODIERT)	
0	0	REGISTER DIREKT	USER- MODE
0	1	REGISTER INDIREKT	USER- MODE
1	0	REGISTER INDIREKT MIT POSTINKREMENT	USER- MODE
1	1	NICHT BENUTZT	

Fig. 5

4 BIT	REGISTER- SELEKT- CODE (BINÄR CODIERT)
0000	REGISTER R0
0001	REGISTER R1
0010	REGISTER R2
0011	REGISTER R3
0100	REGISTER R4
0101	REGISTER R5
0110	REGISTER R6
0111	REGISTER R7
1000	REGISTER R8
1001	REGISTER R9
1010	REGISTER R10
1011	REGISTER R11
1100	REGISTER R12
1101	REGISTER R13
1110	REGISTER R14
1111	REGISTER STACKPOINTER

Fig. 6

Nummer: DE 198 26 826 A1
Int. Cl.[6]: G 06 F 9/30
Offenlegungstag: 15. Juli 1999

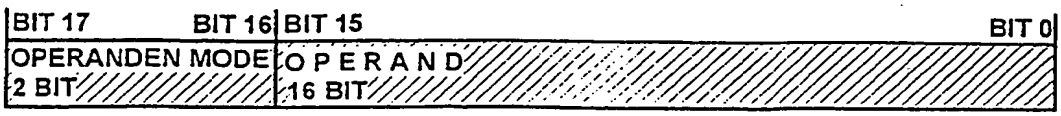

Fig. 7

BIT 17		BIT 16
2 BIT OPERANDEN- MODE (BINÄR CODIERT)		
0	0	REGISTER
0	1	MEMORY- ADRESSE DIREKT
1	0	KONSTANTE DATEN
1	1	NICHT ERLAUBT

Fig. 8

	BIT 15 14	13 10	9	8 7	6 BIT 0
0 0 REGISTER	REGISTER MODE 2 BIT	REGISTER- SELECT 4BIT	TASK SELECT 1 BIT	CONTROL- REGISTER 2 BIT	UNBENUTZT 7 BIT

Fig. 9

2 BIT	REGISTER- ADRESSIERUNGS- MODE	(BINÄR CODIERT)
0 0	REGISTER DIREKT	USER- MODE
0 1	REGISTER INDIREKT	USER- MODE
1 0	REGISTER INDIREKT MIT POSTINKREMENT	USER- MODE
1 1	REGISTER DIREKT	SUPERVISOR- MODE

Fig. 10

Nummer: DE 198 26 826 A1
Int. Cl.[6]: G 06 F 9/30
Offenlegungstag: 15. Juli 1999

4 BIT	REGISTER-SELEKT-CODE (BINÄR CODIERT)
0000	REGISTER R0
0001	REGISTER R1
0010	REGISTER R2
0011	REGISTER R3
0100	REGISTER R4
0101	REGISTER R5
0110	REGISTER R6
0111	REGISTER R7
1000	REGISTER R8
1001	REGISTER R9
1010	REGISTER R10
1011	REGISTER R11
1100	REGISTER R12
1101	REGISTER R13
1110	REGISTER R14
1111	ZUSATZADRESSIERUNG CONTROL-REGISTER

Fig. 11

1 BIT	USER-TASK SELECT (BINÄR CODIERT)
0	USER-TASK 0
1	USER-TASK 1

Fig. 12

2 BIT	CONTROL-REGISTER (BINÄR CODIERT)
0 0	STACKPOINTER
0 1	PROGRAMMCOUNTER
1 0	INSTRUCTIONREGISTER
1 1	STACKPOINTER

Fig. 13

	BIT 15 BIT 0
1 0 MEMORY DIREKT	ABSOLUTE MEMORY-ADRESSE 16 BIT

Fig. 14

	BIT 15 BIT 0
1 1 KONSTANTE	DATEN-KONSTANTE 16 BIT (DATENBIT 15... 0)

Fig. 15